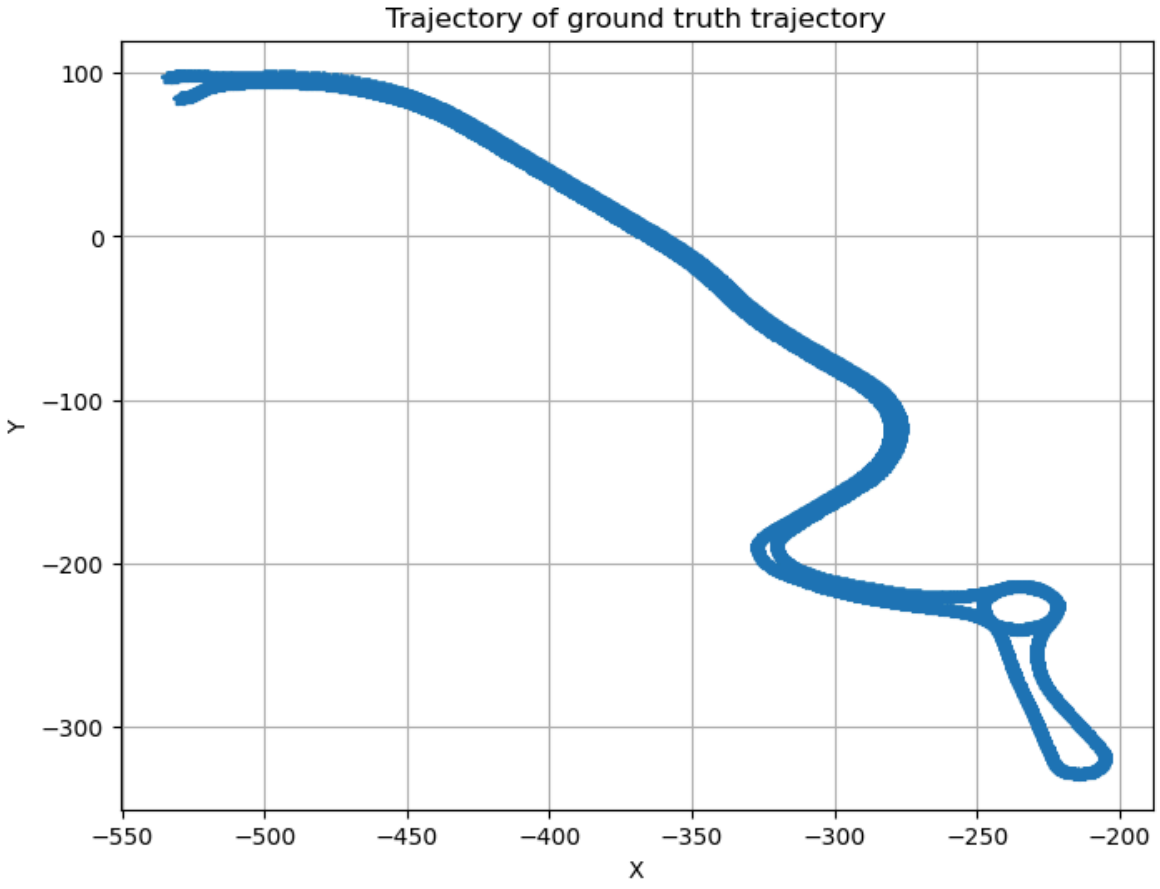
In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import interpolate
```

Question 1.1

In [5]:
```
# Read data from csv
data = pd.read_csv('./waypoints-1.csv', header=None)
data.columns = ['x', 'y']
sampling_interval = 1 / 30
data['time'] = data.index * sampling_interval
```

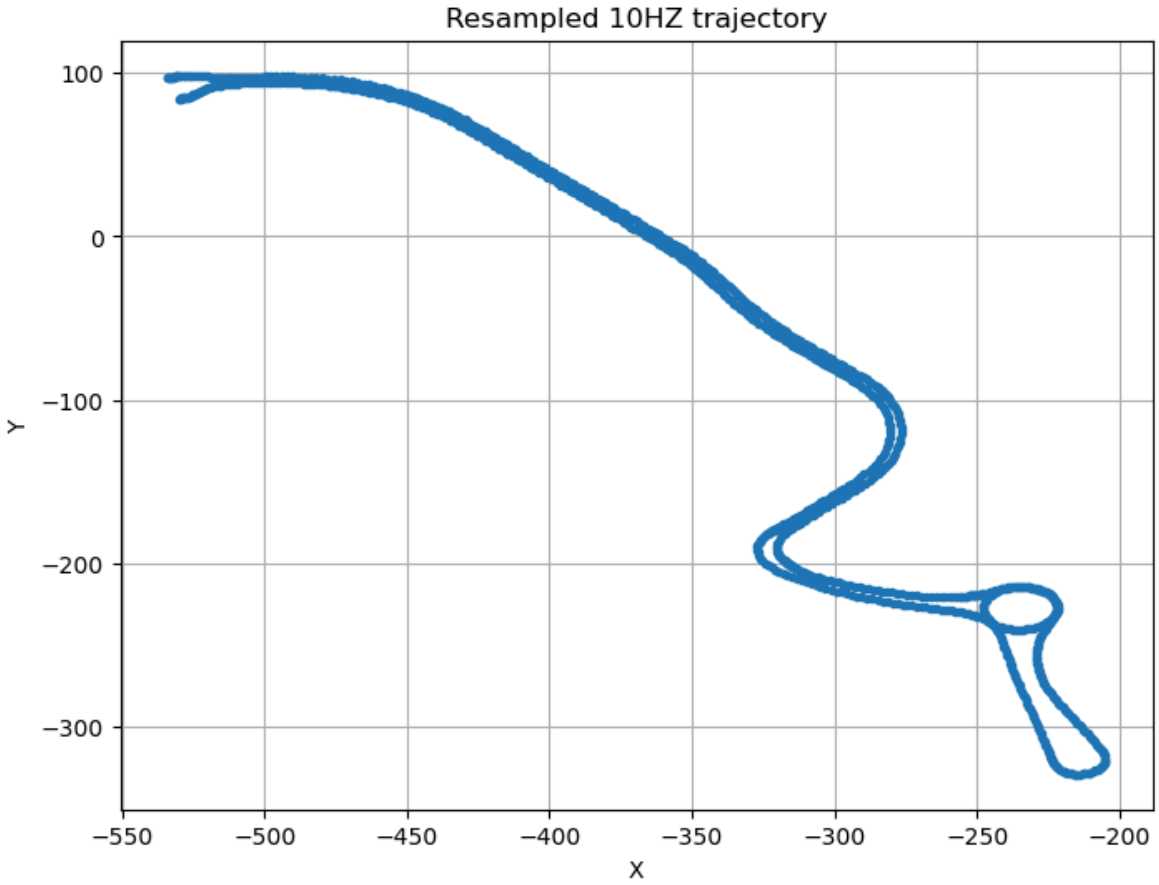
In [8]:
```
x = data['x']
y = data['y']

# Plot ground truth trajectory
plt.figure(figsize=(8, 6))
plt.plot(x, y, marker='*')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Trajectory of ground truth trajectory')
plt.grid(True)
plt.savefig("True.png")
plt.show()
```

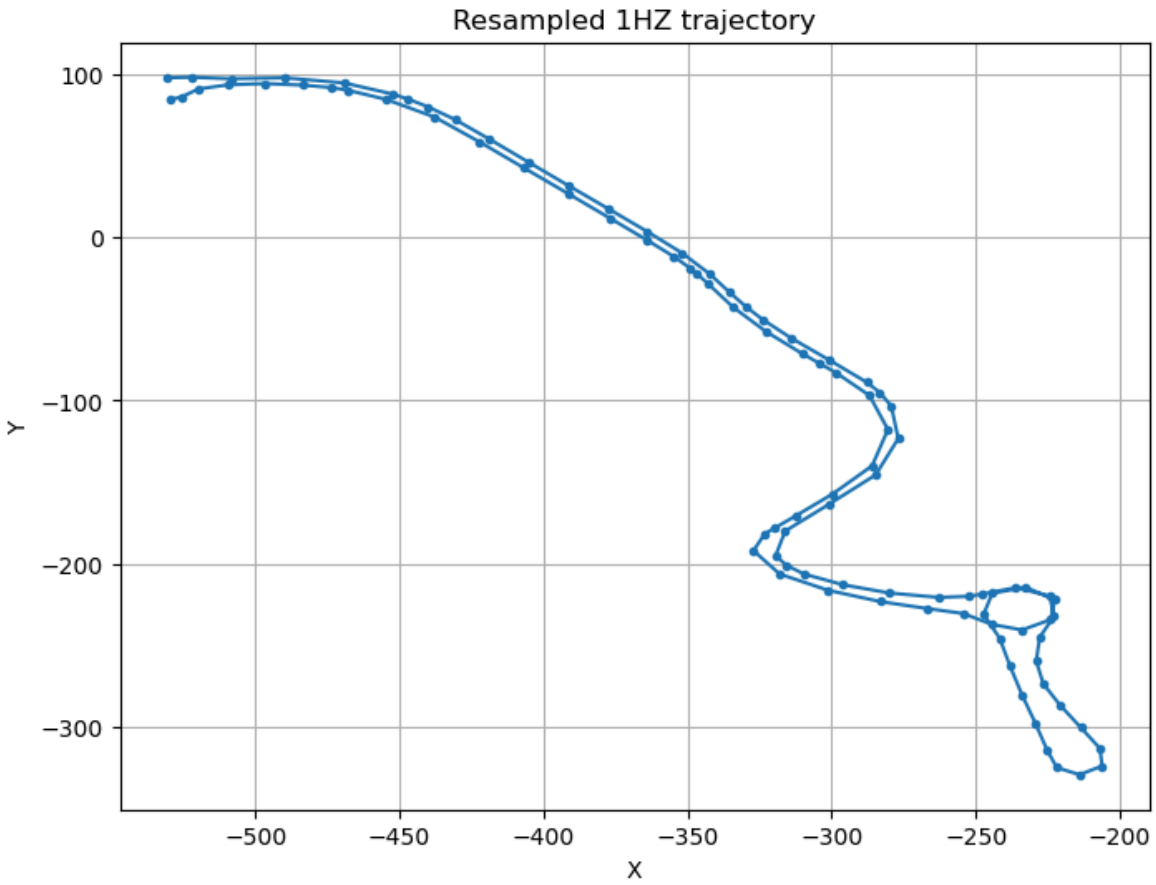


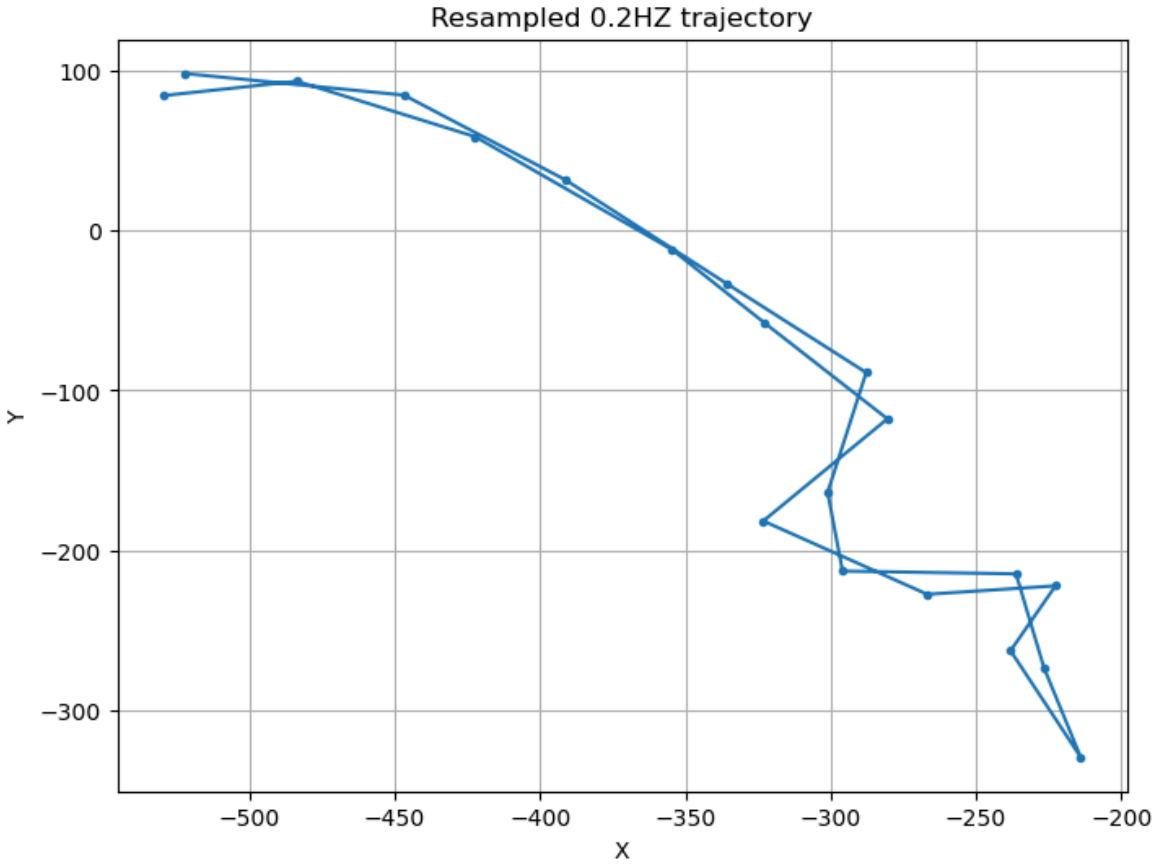
Question 1.2

In [11]:
```python
# Downsample
def downsample_to_frequency(data, original_freq, target_freq):
    step = int(original_freq / target_freq)
    valid_indices = np.arange(0, len(data), step)

    sampled_times = data['time'].iloc[valid_indices].tolist()
    sampled_x = data['x'].iloc[valid_indices].tolist()
    sampled_y = data['y'].iloc[valid_indices].tolist()

    return sampled_times, sampled_x, sampled_y

#Time, X, Y
t_10hz, x_10hz, y_10hz = downsample_to_frequency(data, 30, 10)
t_1hz, x_1hz, y_1hz = downsample_to_frequency(data, 30, 1)
t_2hz, x_2hz, y_2hz = downsample_to_frequency(data, 30, 0.2)
```

In [15]:
```python
# Plot trajectory for downsampled data
def plot_downsample(x, y , freq, tag):
    non_nan_data = data.copy().dropna(subset=['y'])
    plt.figure(figsize=(8, 6))
    linestyle = '-' if tag == 'Resampled' else ''
    plt.plot(x, y, marker='o', linestyle=linestyle,markersize=3)
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title(f'{tag} {freq}HZ trajectory')
    plt.grid(True)
    plt.savefig(f"{tag}_{freq}.png")
    plt.show()
    return

plot_downsample(x_10hz,y_10hz, '10', 'Resampled')
plot_downsample(x_1hz,y_1hz, '1', 'Resampled')
plot_downsample(x_2hz,y_2hz, '0.2', 'Resampled')
```


Resampled 10HZ trajectory
100
0
−100
−200
−300
Y
−550
−500
−450
−400
−350
−300
−250
−200
X


Resampled 1HZ trajectory
100
0
−100
−200
−300
Y
−500
−450
−400
−350
−300
−250
−200
X

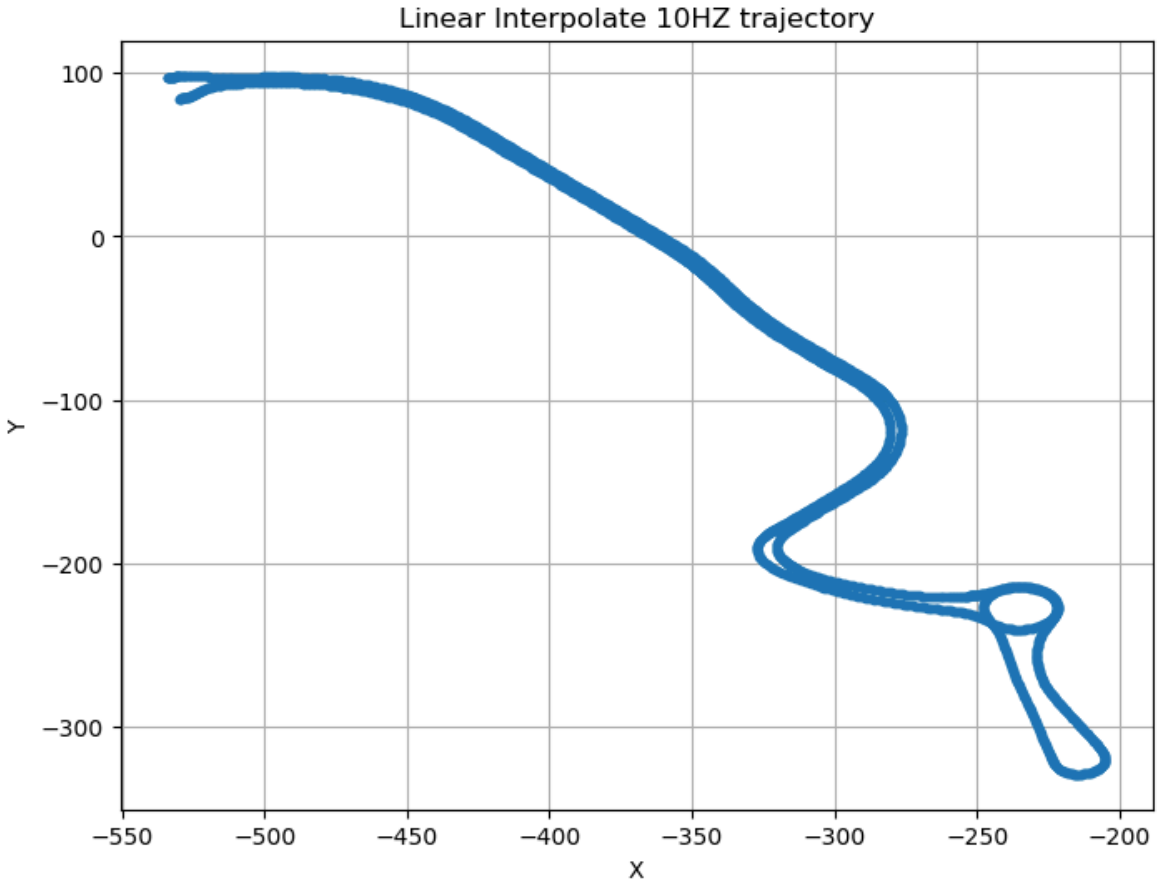


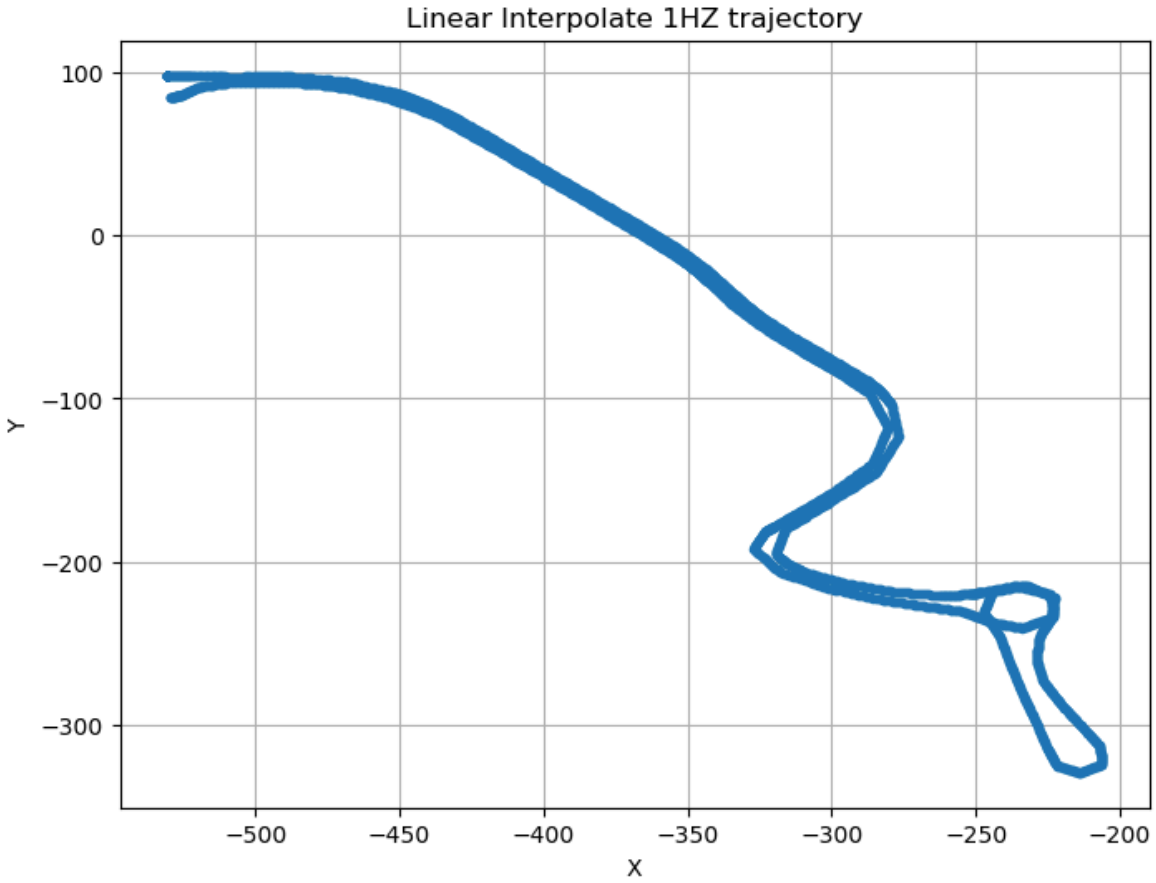
Question 1.2 a

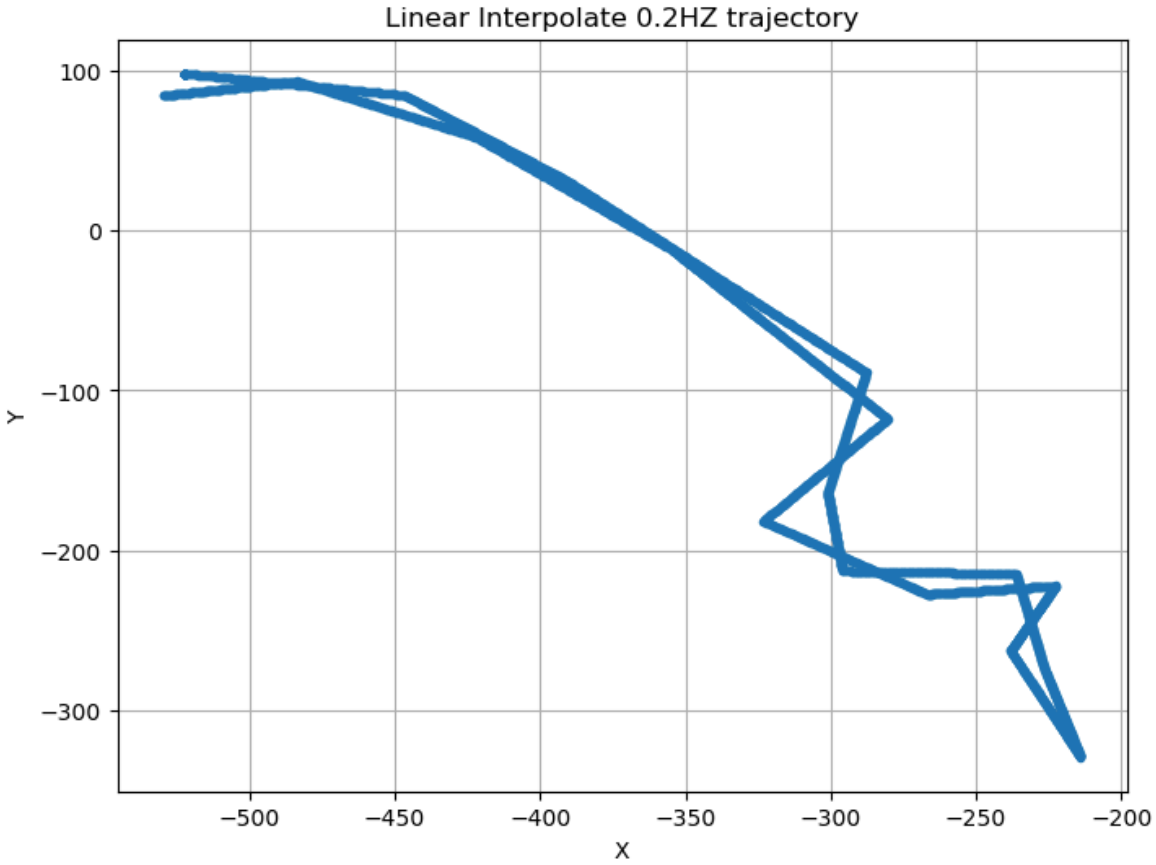
In [17]:
```
# Linear Interpolation
def linear_interpolate(original_data, t, x, y):
    original_time = np.array(original_data['time'].tolist())
    t_new = np.setdiff1d(original_time, t)

    # Apply Linear Interpolation
    x_new = np.interp(t_new, t, x).tolist()
    y_new = np.interp(t_new, t, y).tolist()

    # Compute Euclidean distance
    indices = np.searchsorted(original_time, t_new)
    x_ground_truth = original_data['x'].iloc[indices].values
    y_ground_truth = original_data['y'].iloc[indices].values
    errors = np.sqrt((x_new - x_ground_truth) ** 2 + (y_new - y_ground_truth) **
    cumulative_error = np.cumsum(errors)
    x_new = x_new + x
    y_new = y_new + y
    return x_new, y_new, cumulative_error[-1]

x_2hz_linear, y_2hz_linear, err2_linear = linear_interpolate(data, t_2hz, x_2hz,
x_1hz_linear, y_1hz_linear, err1_linear = linear_interpolate(data, t_1hz, x_1hz,
x_10hz_linear, y_10hz_linear, err10_linear = linear_interpolate(data, t_10hz, x_
print(f'10HZ Cummulative Error of Linear Interpolation: {err10_linear}')
print(f'1HZ Cummulative Error of Linear Interpolation: {err1_linear}')
print(f'0.2HZ Cummulative Error of Linear Interpolation: {err2_linear}')
```

```
10HZ Cummulative Error of Linear Interpolation: 45.77082837475039
1HZ Cummulative Error of Linear Interpolation: 1530.6420811644919
0.2HZ Cummulative Error of Linear Interpolation: 24109.30874076987
```

In [19]:
```
plot_downsample(x_10hz_linear,y_10hz_linear, '10', 'Linear Interpolate')
plot_downsample(x_1hz_linear,y_1hz_linear, '1', 'Linear Interpolate')
plot_downsample(x_2hz_linear,y_2hz_linear, '0.2', 'Linear Interpolate')
```





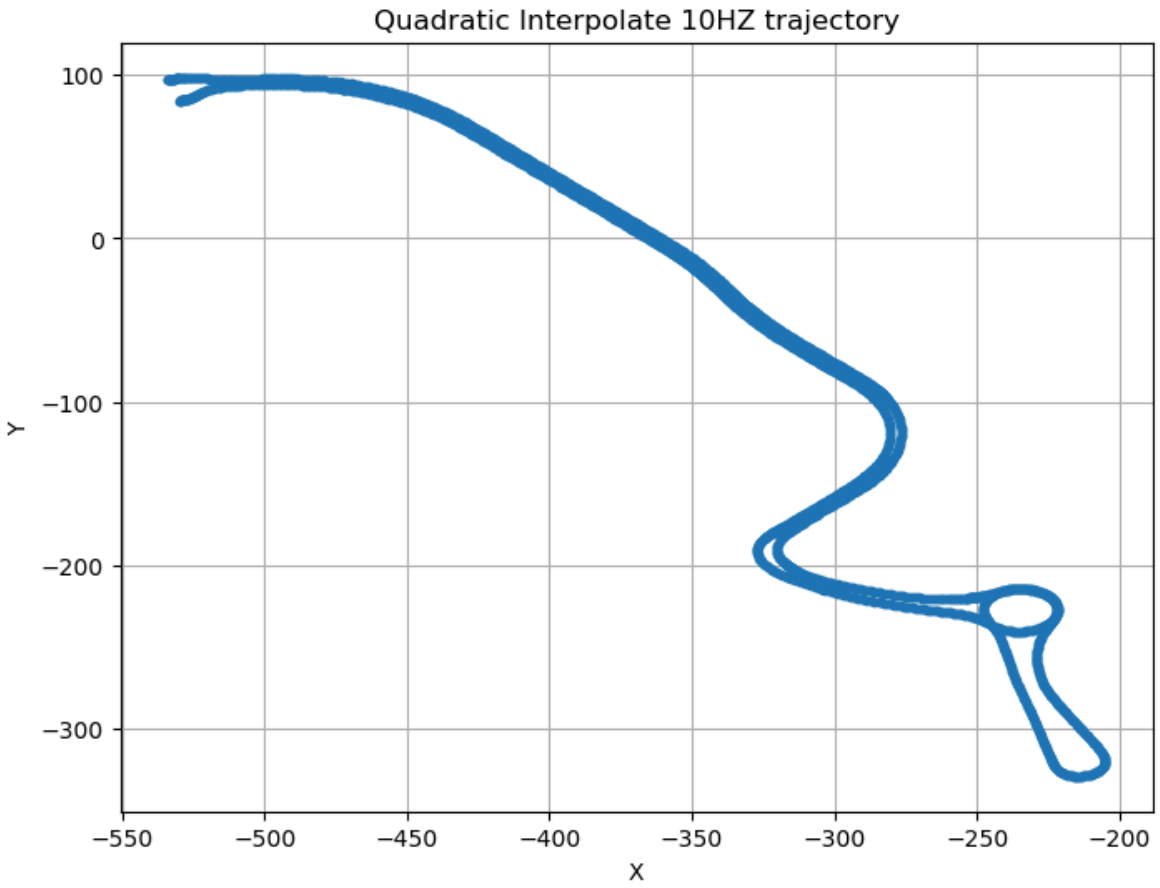


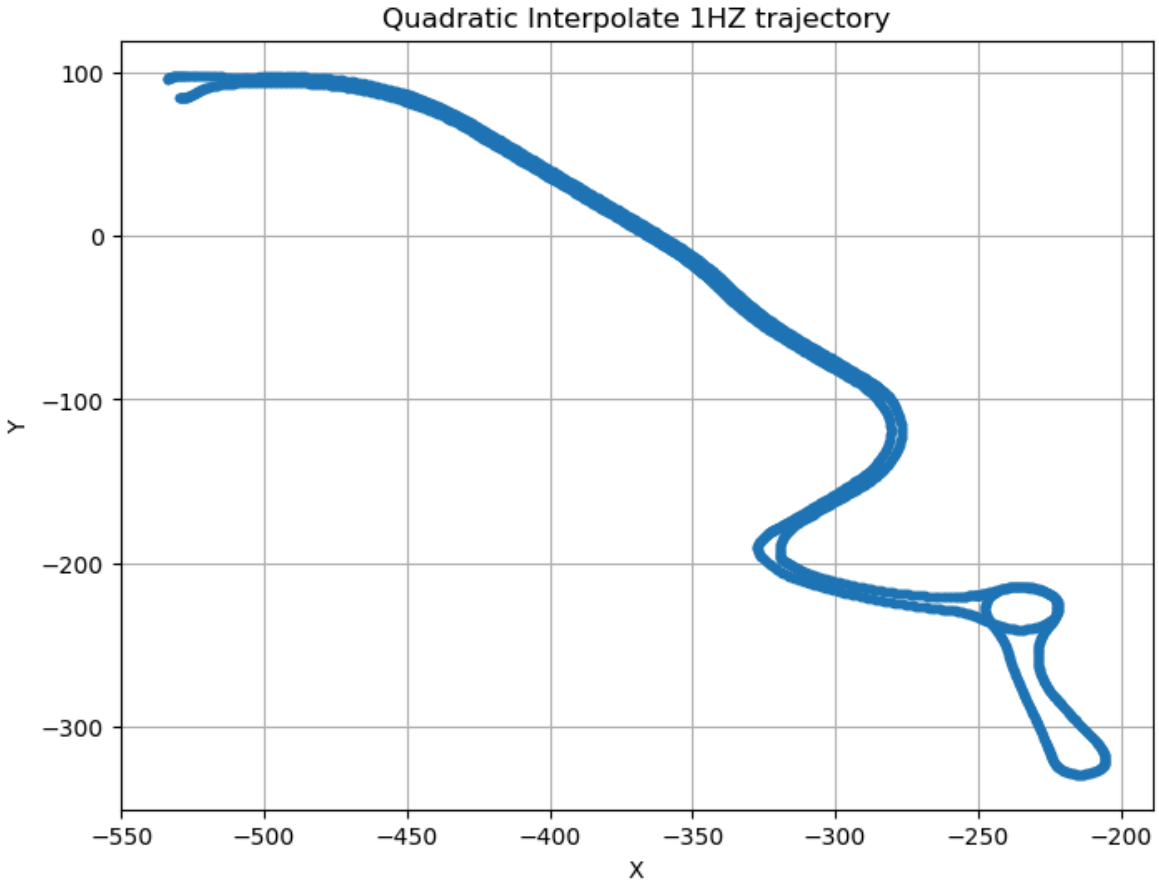
Question 1.2 b

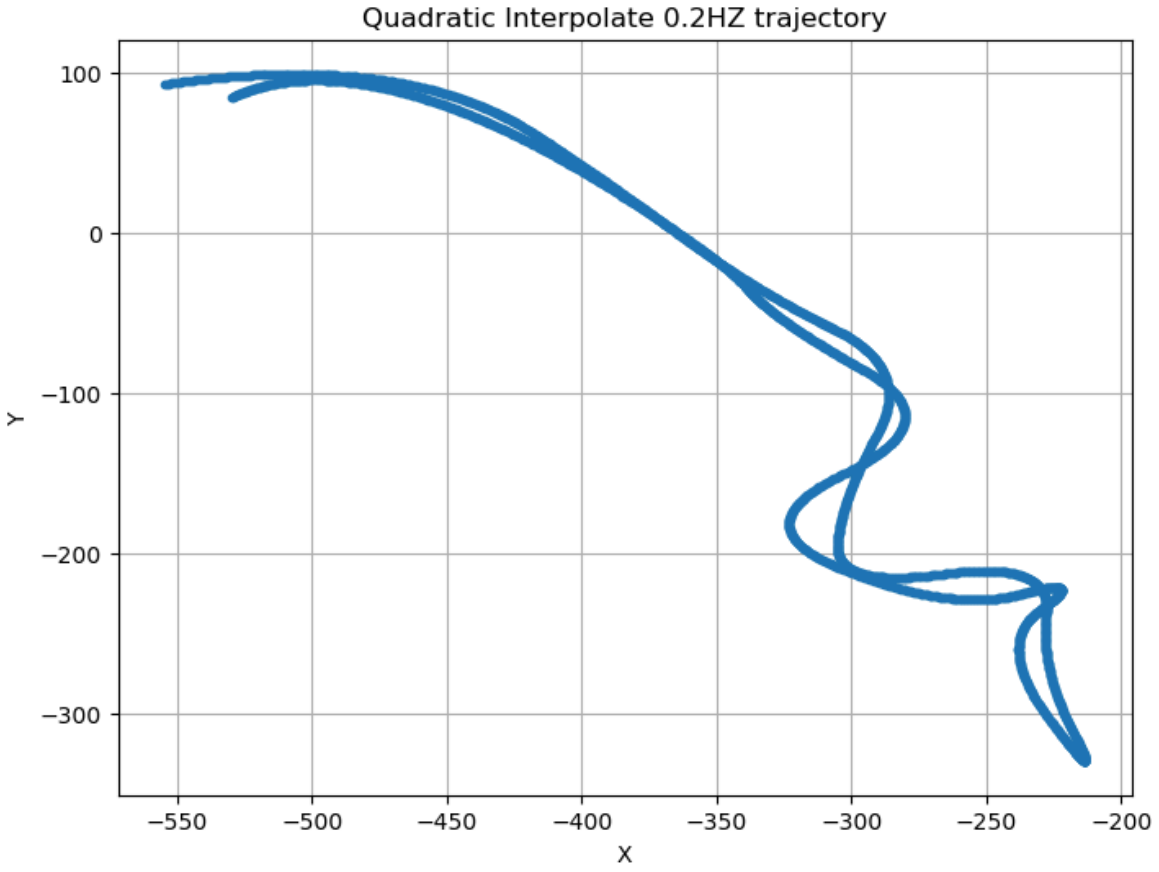
In [21]:
```
# Quadratic Interpolation
def quadratic_interpolate(original_data, t, x, y):
    original_time = np.array(original_data['time'].tolist())
    t_new = np.setdiff1d(original_time, t)

    # Apply Quadratic Interpolation
    px = interpolate.interp1d(t, x, kind='quadratic',fill_value="extrapolate")
    py = interpolate.interp1d(t, y, kind='quadratic',fill_value="extrapolate")
    x_new = px(t_new).tolist()
    y_new = py(t_new).tolist()

    # Compute Euclidean distance
    indices = np.searchsorted(original_time, t_new)
    x_ground_truth = original_data['x'].iloc[indices].values
    y_ground_truth = original_data['y'].iloc[indices].values
    errors = np.sqrt((np.array(x_new) - x_ground_truth) ** 2 + (np.array(y_new)
    cumulative_error = np.cumsum(errors)
    x_new = x_new + x
    y_new = y_new + y
    return x_new, y_new, cumulative_error[-1]

x_2hz_q, y_2hz_q, err2_q = quadratic_interpolate(data, t_2hz, x_2hz, y_2hz)
x_1hz_q, y_1hz_q, err1_q = quadratic_interpolate(data, t_1hz, x_1hz, y_1hz)
x_10hz_q, y_10hz_q, err10_q = quadratic_interpolate(data, t_10hz, x_10hz, y_10hz
print(f'10HZ Cummulative Error of Quadratic Interpolation: {err10_q}')
print(f'1HZ Cummulative Error of Quadratic Interpolation: {err1_q}')
print(f'0.2HZ Cummulative Error of Quadratic Interpolation: {err2_q}')
```

```
10HZ Cummulative Error of Quadratic Interpolation: 45.37993363557641
1HZ Cummulative Error of Quadratic Interpolation: 520.8404259101981
0.2HZ Cummulative Error of Quadratic Interpolation: 19260.701839123296
```

In [23]:
```
plot_downsample(x_10hz_q,y_10hz_q, '10', 'Quadratic Interpolate')
plot_downsample(x_1hz_q,y_1hz_q, '1', 'Quadratic Interpolate')
plot_downsample(x_2hz_q,y_2hz_q, '0.2', 'Quadratic Interpolate')
```







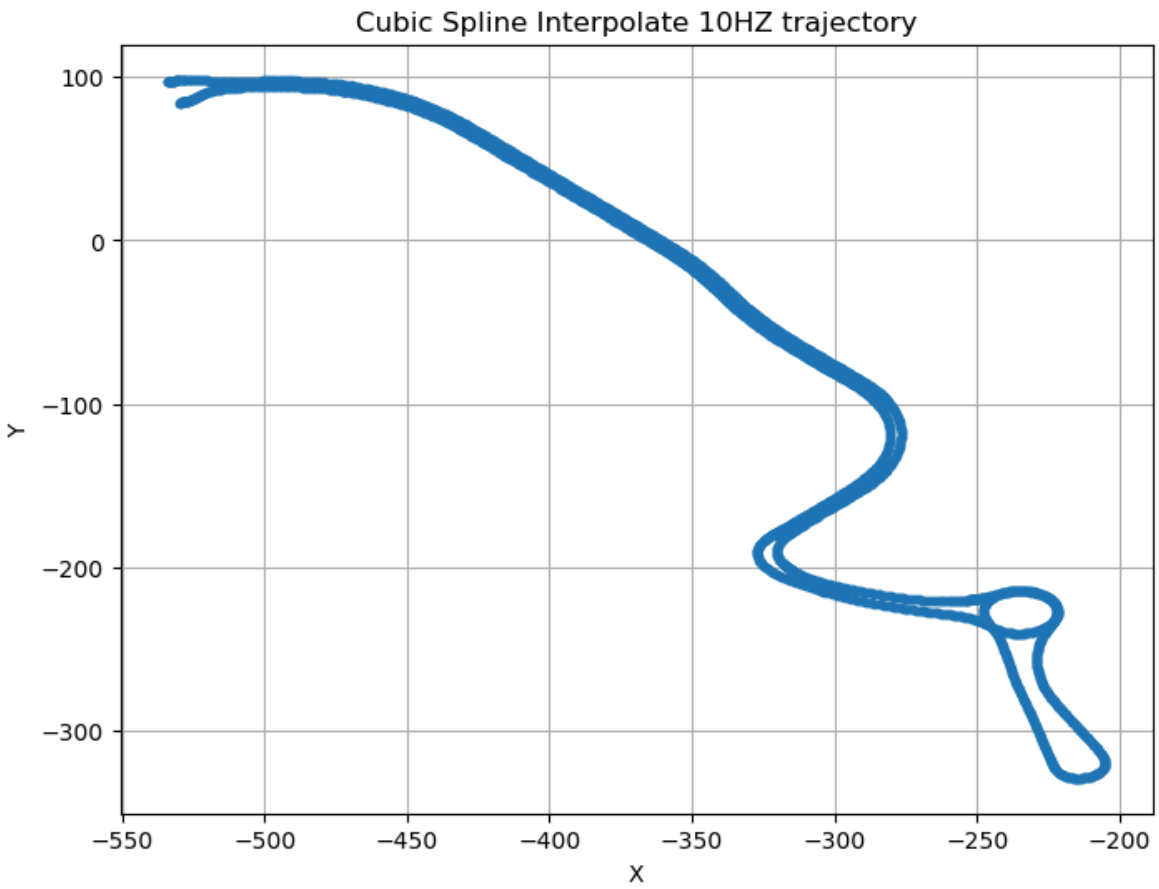
Question 1.2 c

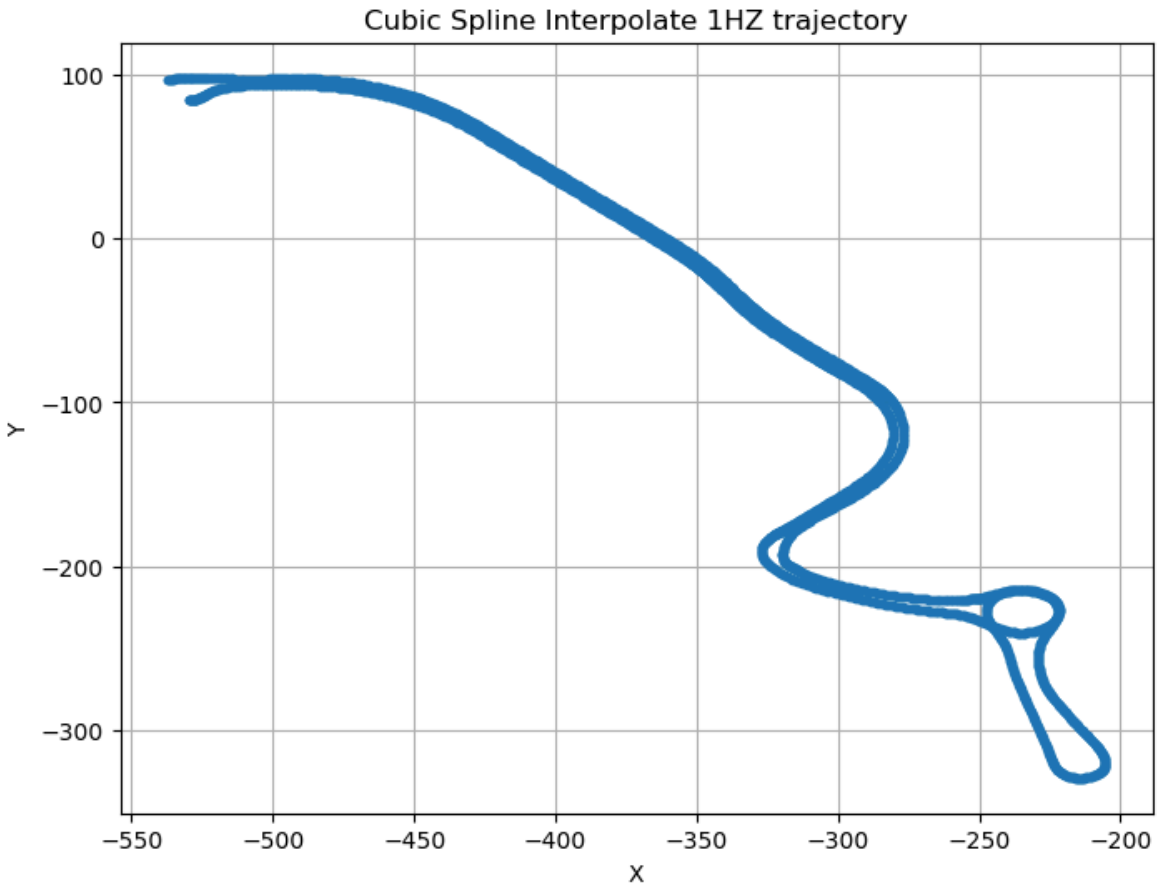
In [29]:
```
# Cubic Spline Interpolation
import numpy as np
```

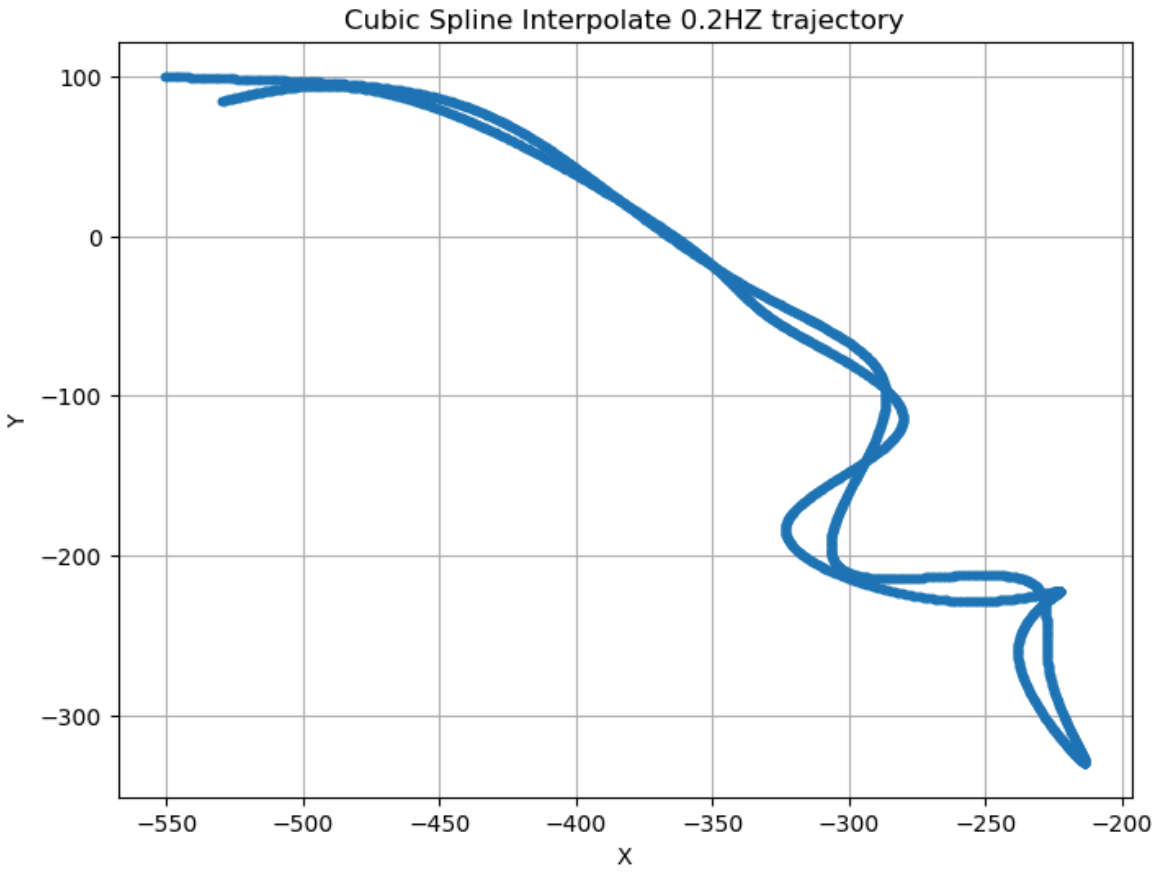
```
from scipy.interpolate import CubicSpline

def cubic_interpolate(original_data, t, x, y):
    original_time = np.array(original_data['time'].tolist())
    t_new = np.setdiff1d(original_time, t)

    # Estimate dx/dt
    dx_dt_start = (x[1] - x[0]) / (t[1] - t[0])
    dx_dt_end = (x[-1] - x[-2]) / (t[-1] - t[-2])

    # Estimate dy/dt
    dy_dt_start = (y[1] - y[0]) / (t[1] - t[0])
    dy_dt_end = (y[-1] - y[-2]) / (t[-1] - t[-2])

    # Apply Cubic Spline Interpolation
    #cs_x = CubicSpline(t, x, bc_type=((1, dx_dt_start), (1, dx_dt_end)))
    #cs_y = CubicSpline(t, y, bc_type=((1, dy_dt_start), (1, dy_dt_end)))
    #cs_x = CubicSpline(t, x)
    #cs_y = CubicSpline(t, y)
    cs_x = CubicSpline(t, x, bc_type='natural')
    cs_y = CubicSpline(t, y, bc_type='natural')
    x_new = cs_x(t_new).tolist()
    y_new = cs_y(t_new).tolist()

    # Compute Euclidean distance
    indices = np.searchsorted(original_time, t_new)
    x_ground_truth = original_data['x'].iloc[indices].values
    y_ground_truth = original_data['y'].iloc[indices].values
    errors = np.sqrt((np.array(x_new) - x_ground_truth) ** 2 + (np.array(y_new)
    cumulative_error = np.cumsum(errors)
    x_new = x_new + x
    y_new = y_new + y

    return x_new, y_new, cumulative_error[-1]

x_2hz_c, y_2hz_c, err2_c = cubic_interpolate(data, t_2hz, x_2hz, y_2hz)
x_1hz_c, y_1hz_c, err1_c = cubic_interpolate(data, t_1hz, x_1hz, y_1hz)
x_10hz_c, y_10hz_c, err10_c = cubic_interpolate(data, t_10hz, x_10hz, y_10hz)
print(f'10HZ Cummulative Error of Cubic Interpolation: {err10_c}')
print(f'1HZ Cummulative Error of Cubic Interpolation: {err1_c}')
print(f'0.2HZ Cummulative Error of Cubic Interpolation: {err2_c}')
```

```
10HZ Cummulative Error of Cubic Interpolation: 45.92407605233177
1HZ Cummulative Error of Cubic Interpolation: 498.7414404415983
0.2HZ Cummulative Error of Cubic Interpolation: 18439.79645738146
```

In [31]:
```
plot_downsample(x_10hz_c,y_10hz_c, '10', 'Cubic Spline Interpolate')
plot_downsample(x_1hz_c,y_1hz_c, '1', 'Cubic Spline Interpolate')
plot_downsample(x_2hz_c,y_2hz_c, '0.2', 'Cubic Spline Interpolate')
```


Cubic Spline Interpolate 10HZ trajectory
100
0
−100
−200
−300
Y
−550
−500
−450
−400
−350
−300
−250
−200
X


Cubic Spline Interpolate 1HZ trajectory
100
0
−100
−200
−300
Y
−550
−500
−450
−400
−350
−300
−250
−200
X


Cubic Spline Interpolate 0.2HZ trajectory
100
0
−100
−200
−300
Y
−550
−500
−450
−400
−350
−300
−250
−200
X

In [ ]: